sciendo

Műszaki Tudományos Közlemények vol. 17. (2022) 60–64.
DOI: Magyar: https://doi.org/10.33895/mtk-2022.17.13
Angol: https://doi.org/10.33894/mtk-2022.17.13

EME MTSZ MTK

KOLLABORATÍV ROBOT ÉS IPARI SZÁMÍTÓGÉP KOMMUNIKÁCIÓS LEHETŐSÉGEINEK VIZSGÁLATA

STUDYING THE COMMUNICATION POSSIBILITIES BETWEEN A COLLABORATIVE ROBOT AND AN INDUSTRIAL COMPUTER

Tóth Szabolcs Balázs,[1] Erdei Timotei István,[2] Husi Géza[3]

Debreceni Egyetem Műszaki Kar, Debrecen, Magyarország

[1] *szabolcs978@gmail.com*

[2] *timoteierdei@eng.unideb.hu*

[3] *husigeza@eng.unideb.hu*

Abstract

This project is based on the study of collaborative robots (hereafter: Cobot) and industrial PCs, including the communication possibilities between them. In the course of the project, a number of possibilities were studied in order to solve the problem and then summarised to select a collaborative robot and an industrial PC. After that, the focus of the project was on the communication options between the two systems: communication via I/O ports and net-work communication (TCP/IP, Modbus/TCP). Finally, the possible solutions were analysed from different points of view (complexity, flexibility, technical aspects, etc.) and communication between the chosen devices was implemented. The implementation was mainly carried out in a simulation environment, but as the project progressed a communication solution was tested on real industrial devices.

Keywords*: KR, Cobot, UR, PLC, IPC, Beckhoff, TCP/IP, Modbus/TCP.*

Összefoglalás

A bemutatásra kerülő projekt alapja, kollaboratív robotok (továbbiakban: KR) és ipari számítógépek, azon belül is a közöttük megvalósítható kommunikációs lehetőségek vizsgálata. A projekt kivitelezése során megvizsgáltam számos lehetőséget a probléma megoldásához, majd ezeket összegezve, egy-egy kollaboratív robotot és ipari számítógépet választottam ki. Mindezek után a két rendszer közötti kommunikációs lehetőségek kerültek a projekt fókuszába: I/O portokon való kommunikáció és hálózati kommunikáció (TCP/IP, Modbus/TCP). Különböző szempontok szerint elemeztem a megoldási lehetőségeket (bonyolultság, rugalmasság, műszaki szempontok stb.) Végül megvalósítottam a választott eszközök közötti kommunikációt. A megvalósítás elsősorban szimulációs környezetben történt, viszont a projekt előrehaladtával valós ipari berendezéseken is teszteltem a kommunikációs megoldásomat.

Kulcsszavak*: KR, Cobot, UR, PLC, IPC, Beckhoff, TCP/IP, Modbus/TCP.*

1. Bevezetés

Az iparban egyre inkább nagyobb hangsúlyt kapnak a robotok, így a munka jelentős részét már nem ember végzi. Viszont vannak még olyan feladatok, ahol emberi beavatkozás is szükséges. Ez az oka annak, hogy KR-t fejlesztettek ki [1].

Egy robottal való kommunikációnak sokféle módja van. A robotokat leggyakrabban vezetékes, vezeték nélküli vagy autonóm módon vezérlik [2].

Célom az volt, hogy valósítsak meg egy kommunikációs megoldást KR és ipari vezérlő között.

2. Kiválasztott ipari vezérlő és robot

Miután megvizsgáltam a kollaboratív robotok és ipari vezérlők főbb tulajdonságait, kiválasztottam egyet-egyet közülük. Vezérlőnek én az **1. ábrá**n látható Beckhoff CX5140 Embedded PC-t választottam. A KR, melyet kiválasztottam, az UR10 Cobot **2. ábra**, mely számos kommunikációs lehetőséggel rendelkezik, többek között támogatja a TCP/IP kommunikációs protokollt is. Mindkét eszköz programozása könnyen elsajátítható, bár az ipari vezérlő igényel némi előismeretet.

1. ábra. *CX5140 Embedded PC* **[3]**

2. ábra. *UR10 Cobot* **[4]**

3. Kommunikációs lehetőségek a kiválasztott rendszerek között

Három kommunikációs lehetőséget vizsgáltam meg. A TCP (transmission control protocol) az az internetes szabvány, amely biztosítja az adatcsomagok sikeres átvitelét az eszközök között a hálózaton keresztül. A TCP az internetes protokollal (IP) együttműködve határozza meg az online adatcsere módját. A két protokollt együttesen TCP/IP-nek nevezik **[5]**.

A Modbus TCP nem más, mint a Modbus RTU üzenet elküldése egy Ethernet csomagba ágyazva, de a soros kapcsolat helyett hálózaton keresztül továbbítjuk az üzenet. A TCP/IP pedig a Modbus TCP-üzenetek átviteli közegét biztosítja. Egyszerűen fogalmazva, a TCP/IP lehetővé teszi az automatizálási eszközök közötti bináris adatblokkok cseréjét **[6]**.

Az I/O modulok a legegyszerűbb módja a kommunikációnak. Feladatom rugalmasságához leginkább a TCP/IP kommunikációs protokoll illik, amely szerver/kliens alapú.

4. TCP/IP-szerver és kliensprogram

A kommunikáció megvalósítása érdekében a két ipari eszköz között a szerver/kliens rendszerben, egy vezetékes LAN-kommunikáció létrehozására volt szükség. Szervernek az ipari vezérlőt választottam, míg kliensnek a robotot. Az eszközök IP-címeinek hálózati szegmensei megegyeznek, a szerver IP-címe: 192.168.2.100, a kliens IP-címe: 192.168.2.113. Ebből adódik, hogy az alhálózati maszk: 255.255.255.0, ez az IPv4 címosztályának a C osztályába tartozik.

A beágyazott PC programját a Beckhoff saját fejlesztőkörnyezetében a TwinCat 3-ban írtam meg ST-nyelven. Az UR10 programját URSim Offline Simulatorában írtam meg, URScript-nyelven.

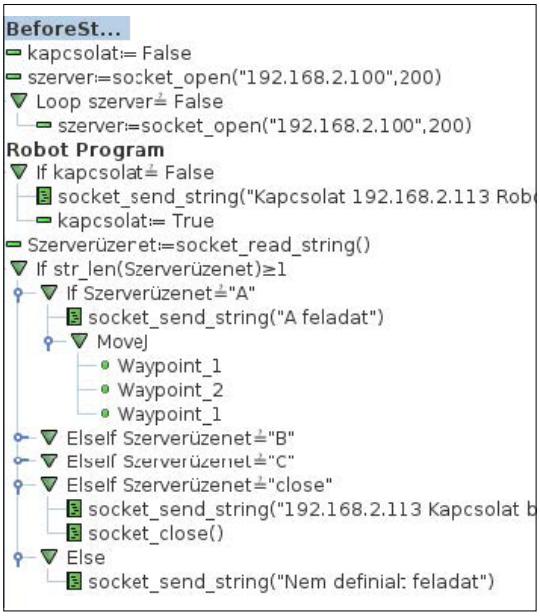
A **3. ábrá**n látható a szerver állapotainak program részlete. A 0. állapot a hallgatói állapot, ilyenkor a szerver működik, és várja a bejövő kapcsolatot. Az 1. állapot sikeres kapcsolódáskor áll fel, majd innen a 2., azaz üzenetcsere-állapotba lép. Üzenetküldés után a 3. állapotba lép a program, majd, a 4. állapotba pedig a szerver lezárását követően lép a program.

A **4. ábrá**n látható a kliensprogram részlete. A kliens megírásánál a BeforeStart szegmensben deklaráltam egy 'kapcsolat' és egy 'szerver' boolean változót. A 'szerver' változó értéke TRUE lesz, amint a kapcsolat létrejön a szerverrel. Majd a sikeres kapcsolat után a kliens üzenetet küld a szervernek, és belép az üzenetküldő/-fogadó állapotba. Előre definiált üzenetek alapján különböző feladatokat tud az UR-kliens végrehajtani, valamint a TCP/IP sajátosságának köszönhetően az üzenetek az elküldés sorrendjében érnek célba.

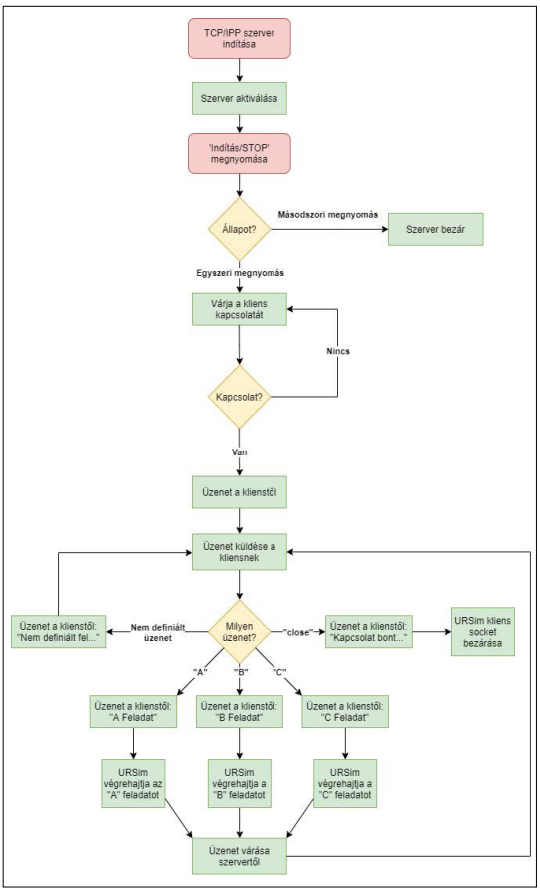
```
CASE nState OF
0: (* Listening State *)
    bReceive := FALSE;
    IF eState = eSOCKET_SUSPENDED THEN
    sMessage := 'Kapcsolatra vár...';
    MEMSET(ADR(sReceivedData),0,SIZEOF(sReceivedData)); (* Delete Received data *)
    ELSE IF eState = eSOCKET_CONNECTED THEN
        nState := 1;
    END_IF
    END_IF
1: (* Connection State *)
    IF NOT fbServer.bBusy THEN
            IF NOT fbServer.bError THEN
                IF eState = eSOCKET_CONNECTED THEN(* Connected *)
                    sMessage := 'Kapcsolat létrejött!';
                    bReceive := TRUE;
                    nState := 2;
                END_IF
            END_IF
    END_IF
2:(* Exchange data State *)
            (* ---------------------- Receive data ---------------------------------- *)

        IF NOT fbSocketReceive.bBusy AND NOT fbSocketReceive.bError THEN
            IF fbSocketReceive.nRecBytes = 0 THEN
                bDisableSendButton := TRUE; //Disable Send Button
                //IF there's no data received, reset the fbSocketReceive
                bReceive := TRUE;
            ELSE
                bDisableSendButton := FALSE; //Unlock Send Button
            END_IF
        END_IF
        (* ---------------------- Send data ---------------------------------- *)
        IF bSEND THEN
            bReceive := FALSE;
            fbSocketSend.bExecute := TRUE;
                MEMSET(ADR(sReceivedData),0,SIZEOF(sReceivedData)); (* Delete Received data *)
            nstate := 3;
            bSEND := FALSE;
        END_IF
3:(* Send data State *)
            MEMSET(ADR(sSentData),0,SIZEOF(sSentData)); (* Delete Sent data *)
            fbSocketSend.bExecute := FALSE;
            IF NOT fbSocketSend.bBusy AND NOT fbSocketSend.bError THEN
                fbSocketReceive.bExecute := TRUE;
                nstate := 1;
        END_IF
4: (* Disconnect *)
    IF bEnable THEN
```

3. ábra. *Szerverprogramrészlet*



4. ábra. *Kliensprogramrészlet*



5. ábra. *Szerver/kliens kapcsolat-folyamatábra*

5. A szerver/kliens kapcsolat működése

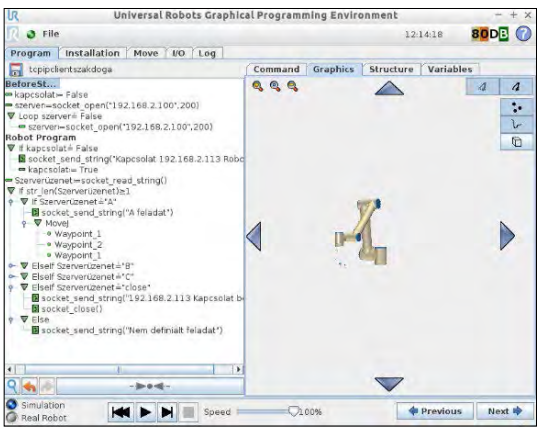
Az **5. ábrá**n látható folyamatábra mutatja be a kapcsolat működését.

Legelőször a szervert állítom be, és indítom el. Majd miután a szerver várja a kliens kapcsolódási szándékát, elindítom a kliensprogramot is, és pár másodpercen belül létre is jön a kapcsolat, amennyiben minden beállítás megfelelő.

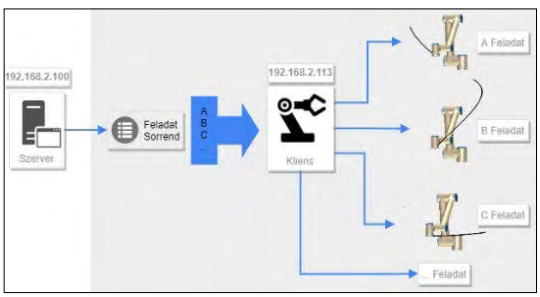
A **6. ábrá**n látható szimulációs környezetben megvalósított TCP/IP kapcsolat. A kapcsolat létrejötte után a szerver képes üzenetet küldeni, illetve fogadni a klienstől. Ez oda-vissza igaz. A **7. ábrá**n láthatók azok a feladatok, amelyeket a robotkliens képes végrehajtani a szerver utasításai alapján.

A kliensprogram itt most háromféle különböző feladatot tud végrehajtani a socket lezárásán és a „nem definiált feladat” üzenet küldésén kívül. Kaphat "A”, "B” vagy "C” üzenetet, és ennek megfelelőképpen hajtja végre az adott feladatot.

Ez a konfiguráció a két eszköz direkt összekapcsolásával jött létre. Viszont bizonyos esetekben router közbeiktatásával fejleszthetjük a kommunikációt, amennyiben több klienssel is szeretnénk egy szerverre csatlakozni.



6. ábra. *Kapcsolat szimulációs környezetben*



7. ábra. *A robot feladatai*

A megvalósított kommunikációt valóságban is leteszteltem a Vitesco debreceni lokációjában található UR10-es cobottal és egy CX5140 ipari vezérlővel.

6. Fejlesztések

A TCP/IP kommunikációs protokoll egyik nagy előnye, hogy könnyedén tudom bővíteni a szerverprogramomat, így akár több klienssel való kapcsolathoz is megírhatom azt. Viszont én most csak 2 klienses megoldást fejlesztettem ki. Ez azért fontos fejlesztés, mert ezzel lehetőség nyílik arra, hogy 2 robot kollaboratív módon egy munkatérben, közös időben tudjon dolgozni egy adott munkafolyamaton. Továbbá a robotok pozícióját be tudom olvasni és el tudom küldeni a másik robotnak. Ezt a fejlesztést továbbgondolva egy olyan kollaboráció alakul ki, amelyben nem csak az ember dolgozik együtt a robottal, hanem robot a robottal is azáltal, hogy képes a két eszköz egymással kommunikálni egy köztes TCP/IP szerveren keresztül. Így még biztonságosabban történhet az ipari munka.

A **8. ábrá**n látható egyszerű Pick and Place-feladatot ellátó 2 KR az operátorral együtt dolgozik. A TCP/IP kommunikációs megoldás adta lehetőségeknek köszönhetően gyors adatcsere mehet végbe a két dolgozó robot között, illetve a szerver HMI-jén keresztül az operátor is képes nyomom követni a feladatot.

A **9. ábrá**n látható szerver HMI-je lehetővé teszi a kezelő számára, hogy különböző utasításokat adjon a kliens számára.

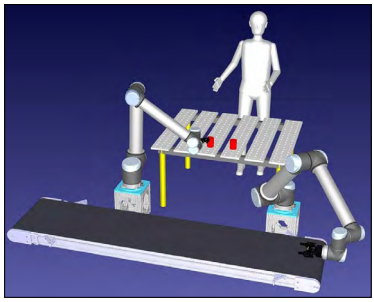
A **10. ábrá**n látható 2. kliens kezelőfelületéről utasításokkal lekérdezhetjük a robot pozícióját, majd később a rögzített állapotba vissza is vezérelhetjük a robotot. Ez a megoldás további lehetőségek számára ad teret, melyek kivitelezését későbbi tanulmányaim során tervezem.

7. Következtetések

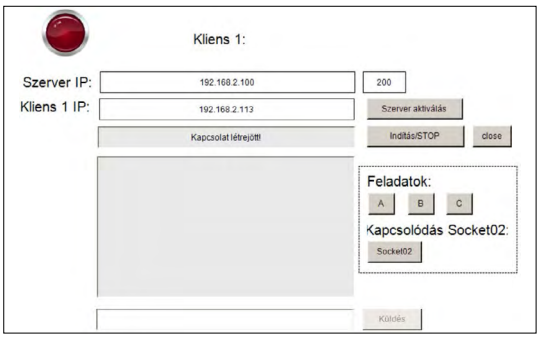
A projekt során kitűzött célok teljesültek, és ezek a projektbeszámolóban kifejtésre kerültek. A kommunikációs megoldás megvalósult szimulációs környezetben, illetve valós ipari berendezéseken is ki lett próbálva. Továbbfejlesztésként pedig egy biztonságosabb, kollaboratív tér megvalósítását realizáltam.

Köszönetnyilvánítás

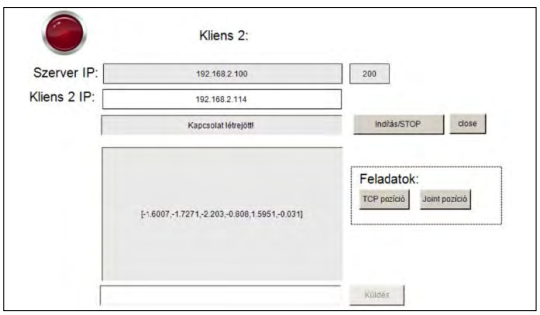
Ezúton szeretnék köszönetet mondani Kamrás Ádám Beckhoff szoftvermérnöknek, hogy segítette munkámat. Kamrás Ádám szakmai tudásával nagy segítségem volt a Beckhoff IPC-k működésének megértésében.



8. ábra. *Kollaboráció ember és robot között*



9. ábra. *Szerver-HMI, 1.kliens*



10. ábra. *Szerver-HMI, 2.kliens*

Szakirodalmi hivatkozások

[1] Kollaboratív robotok (2021.09.20) https://www.mmk.hu/informaciok/hirek/kollaboratiV-robotok

[2] Kulcsár B.: *Robottechnika.* LSI Oktatóközpont, 1998.

[3] Beckhoff Automation GmbH (2021.09.28) https://www.beckhoff.com/hu-hu/products/ipc/embedded-pcs/cx5100-intel-atom/cx5140.html

[4] Universal Robots (2021.09.28) https://www.universal-robots.com/cb3

[5] Casad J.: *TCP/IP in 24 Hours*. Sams Teach Yourself, Pearson Education (US), 2017.

[6] Real Time Automation (2021.10.04) https://www.rtautomation.com/technologies/modbus-tcpip